Week 12: Capstone Project Part 5.2 NUS Salman

I do not have paid Open AI for APIs , so using Hugging Face

```
!pip install -q transformers accelerate
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 115.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 65.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 53.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 6.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 14.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 8.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 6.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 102.1 MB/s eta 0:00:00
```

```
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline

model_id = "defog/sqlcoder-7b-2"

tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    device_map="auto",
    trust_remote_code=True
)

text2sql = pipeline("text-generation", model=model, tokenizer=tokenizer)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json:  1.84k/?  [00:00<00:00,  40.9kB/s]

tokenizer.model:  100%  500k/500k  [00:00<00:00,  3.87MB/s]

tokenizer.json:  1.84M/?  [00:00<00:00,  17.8MB/s]

special_tokens_map.json:  100%  515/515  [00:00<00:00,  15.7kB/s]

config.json:  100%  691/691  [00:00<00:00,  16.0kB/s]

model.safetensors.index.json:  23.9k/?  [00:00<00:00,  2.05MB/s]

Fetching  3  files:  100%  3/3  [03:28<00:00,  86.27s/it]

model-00001-of-00003.safetensors:  100%  4.94G/4.94G  [03:24<00:00,  41.9MB/s]

model-00003-of-00003.safetensors:  100%  3.59G/3.59G  [02:47<00:00,  29.6MB/s]

model-00002-of-00003.safetensors:  100%  4.95G/4.95G  [03:27<00:00,  28.5MB/s]

Loading  checkpoint  shards:  100%  3/3  [01:01<00:00,  19.51s/it]

generation_config.json:  100%  111/111  [00:00<00:00,  12.6kB/s]

WARNING:accelerate.big_modeling:Some parameters are on the meta device because they were offloaded to the cpu and disk.
Device set to use cuda:0
```

```
import sqlite3

def setup_database():
    conn = sqlite3.connect("company.db")
    c = conn.cursor()

    c.execute('''
        CREATE TABLE IF NOT EXISTS employees (
            id INTEGER PRIMARY KEY,
            name TEXT,
            department TEXT,
            salary REAL
        )
    ''')
    c.execute('''
        CREATE TABLE IF NOT EXISTS departments (
            id INTEGER PRIMARY KEY,
            name TEXT,
            budget REAL
        )
    ''')
    c.execute("INSERT OR IGNORE INTO employees VALUES (1, 'John Doe', 'Engineering', 75000)")
    c.execute("INSERT OR IGNORE INTO employees VALUES (2, 'Jane Smith', 'Marketing', 65000)")
    c.execute("INSERT OR IGNORE INTO departments VALUES (1, 'Engineering', 1000000)")
    c.execute("INSERT OR IGNORE INTO departments VALUES (2, 'Marketing', 500000)")
    conn.commit()
    conn.close()

setup_database()
```

```
def generate_sql_from_nl(question):
    prompt = f"""### SQLite SQL tables, with their properties:
# employees(id, name, department, salary)
# departments(id, name, budget)
### A query to answer: {question}
SELECT"""
    result = text2sql(prompt, max_new_tokens=128, do_sample=False)[0]["generated_text"]
    return "SELECT" + result.strip().split("SELECT", 1)[-1].split(";")[0] + ";"
```

```
def validate_sql(sql):
    sql_lower = sql.lower()
    if any(bad in sql_lower for bad in ["drop", "delete", "insert", "update"]):
        raise ValueError("❌ Unsafe query blocked.")
    return sql

def execute_sql(sql):
    conn = sqlite3.connect("company.db")
    cursor = conn.cursor()
```

```
    try:
        validate_sql(sql)
        cursor.execute(sql)
        return cursor.fetchall()
    except Exception as e:
        return [("Error", str(e))]
    finally:
        conn.close()

def format_results(results):
    if not results:
        return "No results found."
    return "\n".join(["\t".join(map(str, row)) for row in results])
```

```
def sql_agent(question):
    print(f"🧠 Question: {question}")
    sql = generate_sql_from_nl(question)
    print(f"📝 Generated SQL: {sql}")
    result = execute_sql(sql)
    print("📊 Result:")
    return format_results(result)

# Run test cases
test_questions = [
    "List all employees.",
    "Which departments have a budget over 750000?",
    "Show employees earning more than 70000",
    "DROP TABLE employees"  # should be blocked
]

for q in test_questions:
    print(sql_agent(q))
    print("-" * 50)
```

```
The following generation flags are not valid and may be ignored: ['temperature']. Set `TRANSFORMERS_VERBOSITY=info` for more details.
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
🧠 Question: List all employees.
The following generation flags are not valid and may be ignored: ['temperature']. Set `TRANSFORMERS_VERBOSITY=info` for more details.
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
📝 Generated SQL: SELECT e.id, e.name, d.name AS department_name, e.salary FROM employees e JOIN departments d ON e.department = d.id;
📊 Result:
No results found.
--------------------------------------------------
🧠 Question: Which departments have a budget over 750000?
The following generation flags are not valid and may be ignored: ['temperature']. Set `TRANSFORMERS_VERBOSITY=info` for more details.
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
📝 Generated SQL: SELECT d.name FROM departments d WHERE d.budget > 750000 ORDER BY d.name NULLS LAST;
📊 Result:
Engineering
--------------------------------------------------
🧠 Question: Show employees earning more than 70000
The following generation flags are not valid and may be ignored: ['temperature']. Set `TRANSFORMERS_VERBOSITY=info` for more details.
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
📝 Generated SQL: SELECT e.name, d.name AS department_name, e.salary FROM employees e JOIN departments d ON e.department = d.id WHERE e.salary > 70000;
📊 Result:
No results found.
--------------------------------------------------
🧠 Question: DROP TABLE employees
📝 Generated SQL: SELECT * FROM employees e;
📊 Result:
1       John Doe        Engineering     75000.0
2       Jane Smith      Marketing       65000.0
--------------------------------------------------
```